Configuration Manual

MSc Research Project
Data Analytics

Anwesha Karmakar
Student ID: x21195838

School of Computing
National College of Ireland

Supervisor: Paul Stynes, William Clifford, Eugene McLaughlin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	…….Anwesha Karmakar……………………………………………………………………
Student ID:	……X21195838………………………………………………………………………….……
Programme:	……MSc Data Analytics……………………………… **Year:** ……2023………………..
Module:	……MSc Research Project………………………………………………………….………
Lecturer:	……Paul Stynes, William Clifford, Eugene McLaughlin ……..…………….………
Submission Due Date:	……14/08/2023……………………………………………………………………….………
Project Title:	……Configuration Manual……………………………………………………….………
Word Count:	……410…………………………… **Page Count:** ……3.……………………….…….………

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Anwesha Karmakar…………………………………………………………………

Date: ……14/08/2023…………………………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Anwesha Karmakar
Student ID: x21195838

1 Introduction

The following configuration manual provides description and procedure of running code scripts to reproduce the research work. It also includes the hardware configuration of the system used for implementation.

2 Development Environment

The development environment used for the research work was python 3.8.10 with keras v2.11.0 and TensorFlow v2.9.1. The model was run using cloud GPU.

- GPU: 1x NVIDIA A10 GPU
- vRAM per GPU: 24 GB
- vCPUs: 30
- RAM: 200 GiB
- Storage: 1.4 TiB

3 Packages and Modules

Model building and training was performed using Jupyter Notebook in Lambda cloud IDE[1]. Important modules and packages required for successfully running code are nltk for text preprocessing, plotly for data visualization, np_utils for one-hot encoding, Numpy for scientific computation, numexpr 2.7.3 or higher for faster numerical evaluation of numpy, tqdm for progress bar, sklearn and matplotlib for visuatizing the data, transformers for pre-trained transfer learning models, TensorFlow-hub for fine tuning and deploying models. Other than these, some standard packages and modules such as re for regular expressions, random for pseudorandom number generation and os for interaction with operating systems were used from Python library.

4 Dataset Details

The resume corpus dataset used in the research was saved in 'Data' folder which was obtained from GitHub public repository[2]. The data contains a resume_corpus.zip file containing all the resumes in .txt format and their corresponding labels in .lab files. A resume_sample.zip file is present with all the resumes in one text file containing reference id,

[1] https://cloud.lambdalabs.com/instances
[2] https://github.com/florex/resume_corpus

occupation and text resumes separated by “:::”. Another file named normalized_classes contain occupations and their corresponding normalised form.

5 Data Transformation

Data transformation is performed using pre-trained transformer model BERT base uncased from transformers. Figure 1 shows requirements and loading of BERT.

Transfer Learning

```
from tqdm.notebook import tqdm
from transformers import BertTokenizer, TFBertModel

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = TFBertModel.from_pretrained('bert-base-uncased')
```

```
Downloading (…)solve/main/vocab.txt: 100%  232k/232k [00:00<00:00, 1.90MB/s]
Downloading (…)okenizer_config.json: 100%  28.0/28.0 [00:00<00:00, 2.82kB/s]
Downloading (…)lve/main/config.json: 100%  570/570 [00:00<00:00, 67.5kB/s]
Downloading model.safetensors: 100%  440M/440M [00:01<00:00, 403MB/s]
```

Figure 1: Transfer Learning Requirement

The data was tokenized using BERT tokenizer. Input ids and attention masks were created from input and converted to NumPy array for further use. The steps are shown in figure 2.

```
def data_creation(X,Y,token):
    input_ids=[]
    attention_masks=[]
    for sent in tqdm(X):
        dbert_inps=token.encode_plus(str(sent),add_special_tokens = True,max_length =512,pad_to_max_length = True,return_attention_mask = True,truncation=True)
        input_ids.append(dbert_inps['input_ids'])
        attention_masks.append(dbert_inps['attention_mask'])

    input_ids=np.asarray(input_ids)
    attention_masks=np.array(attention_masks)
    Y=np.array(Y)
    print(len(input_ids),len(attention_masks),len(Y))
    return input_ids,attention_masks,Y
```

```
model_use = model
token_use = tokenizer
input_ids,attention_masks,Y = data_creation(X,Y,token_use)
```

```
100%  29035/29035 [08:12<00:00, 63.90it/s]
29035 29035 29035
```

Figure 2: BERT Tokenizer

6 Model Training

In this research, data modelling was performed using deep learning neural network. All the modules required for configuring the model was loaded and training was performed.

```
from tensorflow.keras.layers import LSTM,add
from keras.preprocessing.text import Tokenizer
from keras_preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from keras.utils.np_utils import to_categorical
from keras.callbacks import EarlyStopping
from keras.layers import Dropout,MaxPool1D
from tensorflow.keras.layers import Embedding
from tensorflow.keras.layers import Flatten,RNN
```

Figure 3: Loading Layers from Keras

The deep learning model was fine tuned to reduce overfitting. Fine tuning was performed using Adam optimizer and the value of learning rate was set to 0.00000753. The fine tuning is shown in figure 4.

```
METRICS = [
               'accuracy'
        ]
model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
                filepath='./',
                save_weights_only=True,
                monitor='accuracy',
                mode='max',
                save_best_only=True)
model.compile(
                optimizer=tf.keras.optimizers.Adam(learning_rate=0.00000753),
                loss='categorical_crossentropy',
                metrics=METRICS
            )
history = model.fit(OP, Y,
                            epochs=100,
                            verbose=1,
                            shuffle = True,
                            validation_data=(X_test,y_test),
                           callbacks=[model_checkpoint_callback])
```

Figure 4: Fine Tuning Classification Model